Tracking a retail banking transaction end-to-end

You work in a retail bank and your role is to monitor transactions to look for ways to improve the customer experience. For example, you track average duration or a transaction, the number of steps in a transaction, and the highest number of transactions completed by customer. In your bank, the typical transaction has four steps, so you need searches that will enable you to view what happens at each step. This use case provides a wide variety of measurements for a hypothetical four-step banking transaction.

Data required

- **Business service data** for banking transactions

Procedures

- Splunk recommends that customers look into using data models, report acceleration, or summary indexing when searching across hundreds of GBs of events in a single search. The searches provided here are a good starting point, but depending on your data, search time range, and other factors, more can be done to ensure that they scale appropriately.

- Your typical banking transactions may include more than four steps, and some commands, parameters, and field names in the searches below may need to be adjusted to match your environment. In addition, to optimize the searches shown below, you should specify an index and a time range when appropriate.

► Transactions that did not complete

By monitoring the list of sessions that only made it partially to the end, analysts can make better decisions to improve the application. This search not only gives you an indicator of how many did not finish, but it also provides details about each one that did not finish and how many steps they did finish. This is important for accuracy and also for investigation for improving the system.

```
|sourcetype=<banking transaction data source>
|sort - _time
|stats values(_time) AS time values(action) AS action values(customer) AS customer count BY sessionID
|where (count == 1)
```

- Change the count value in the last search command to see how many transactions made it through 2 or 3 steps instead.
- You can print other fields by using list or values in the stats command.

Counts for terminated transaction that did not finish

For a hypothetical, four-step transaction, it is important to have counts on how many transactions terminated at each step. Ideally, you would want all transactions to complete and have zero counts for all the terminated sub-steps. If the number of transactions terminating at the sub-steps is larger than zero, investigate why those transactions did
Running four different subsearches for each transactions step using the `appendcols` command outputs all results into one report.

### Number of steps completed for all transactions started

For a hypothetical, four-step transaction, it is important to have counts on how many steps were performed for each transaction that started. How many went to two steps, three steps, or completed? This information helps measure the quality of the application as it is best if all transactions complete.

```
sourcetype=<banking transaction data source>
| eval _time=strptime(_time,'%Y/%m/%d %H:%M:%S')
| sort _time
| stats count(eval(action="start")) AS start_count count(eval(action="in-progress")) AS in-progress_count count(eval(action="ending")) AS ending_count count(eval(action="finished")) AS finished_count
```

### Average duration chart for transactions that completed

For a hypothetical, four-step transaction, it is important to calculate how long each completed transaction takes. This information could be used to find out what periods in time performance are slow and to see which transactions may have missed SLAs based on agreed upon SLA numbers.

```
sourcetype=<banking transaction data source>
| eval _time=strptime(_time,'%Y/%m/%d %H:%M:%S')
| sort _time
| stats count values(_time) AS _time first(_time) AS first last(_time) AS last BY sessionID | where (count > 3)
| eval duration=(last - first)
| timechart span=10m fixedrange=F avg(duration)
```

### Outliers in transaction duration

It is important to identify outliers in length of transactions. If a customer duration is above the average by N (in this case 2) times the standard deviation, it is most likely a candidate for a SLA violation and it should be investigated.

```
sourcetype=<banking transaction data source>
| eval _time=strptime(time, "%Y/%m/%d %H:%M:%S")
```
Top customers completing transactions

For a hypothetical, four-step transaction, knowing which customers have completed their transactions the most times gives visibility into who is using the application the most with success. Run the search and view the results in a pie chart.

```
sourcetype=<banking transaction data source>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort _time
| stats count last(action) AS action values(_time) AS _time BY sessionID, customer
| where ((action == "finished") AND (count == 4))
| top customer limit=10
```

Transaction steps seen over time

Seeing how many transactions reach a certain step over time gives a representation of how your applications are performing. This helps in solving SLA issues and provides statistics for the performance of your application.

```
sourcetype=<banking transaction data source>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort _time
| timechart span=10s fixedrange=F count(eval(action="start")) AS start count(eval(action="in-progress")) AS in-progress count(eval(action="ending")) AS ending count(eval(action="finished")) AS finished
```

Average duration in between steps of a transaction

Knowing the average duration between each step of a transaction can help provide fine-grained statistics that can help improve performance and see which steps take the longest. For a four-step transaction, do this times in different subsearches and use appendcols to output all results into one table.

```
sourcetype=<banking transaction data source>
| eval _time=strptime(_time, "%Y/%m/%d %H:%M:%S")
| sort _time
| streamstats global=false window=2 first(epoch) AS first last(epoch) AS last last(action) AS action BY sessionID
| eval duration=last-first
| where duration>=0 AND action="in-progress"
| stats avg(duration) AS Avg-In-Progress-Duration
```

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
If you have the Splunk Sankey Diagram - Custom Visualization app from Splunkbase installed on your Splunk instance, you can save the the search as a Sankey Chart.

---

Next steps

Use the results of these searches to evaluate service and capacity and make business recommendations. When taking action on these results, be sure to follow any industry policies and regulations that are required for compliance.

The Splunk Essentials for the Financial Services Industry app helps you automate the searches provided in this article. The app also provides more insight on how they can be applied in your environment, how they work, the difficulty level, and what data can be valuable to run them successfully. In addition, the Splunk Essentials for the Financial Services Industry app provides a number of other monitoring and reporting solutions for financial services:

- ATM usage statistics
- Wire transfer statistics
- Credit card statistics

You might also be interested in monitoring banking transactions for potentially fraudulent activity.

Finally, these additional Splunk resources might help you understand and implement this use case:

- Conf Talk: 40 ways to use Splunk in financial services